☆ Distance Between Two Points

You have to create two classes, *Point2D,* the super class, and its derived class, *Point3D*.
In the given *main* method, we are parsing six values that represent point coordinates. Here, x_1, y_1, z_1 represent the coordinates of the *first* point, and x_2, y_2, z_2 represent the coordinates of the *second* point. You have to implement the two classes and their required methods so that the given main method prints the *2D,* as well as the *3D* distance between the two points.

$$\text{2D } distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$
$$\text{3D } distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Given below are the details of the classes:

Point2D	
x y	Instance variable to store X coordinate Instance variable to store Y coordinate
Point2D(...)	Parameterized constructors to initialize instance variables
double dist2D(Point2D p)	Calculates 2D distance between the current Point2D object and Point2D object passed as parameter
void printDistance(double d)	Print the integer 2D distance between two points

Point3D	

z	Instance variable to store Z coordinate
Point3D(...)	Parameterized constructors to initialize instance variables
double dist3D(Point3D p)	Calculates 3D distance between the current Point3D object and Point3D object passed as parameter
void printDistance(double d)	Prints the integer 3D distance between two points

Input Format

The input contains six lines. Each line contains an integer.
The first three values represent the first coordinate: x_1, y_1, z_1 respectively.
The last three values represent the second coordinate: x_2, y_2, z_2 respectively.

Output Format

The first line contains the *2D* distance between the points.
The next line contains the *3D* distance between the points.

Note: In the display function, print the result rounded UP to the nearest integer.

Sample Input 0

```
1
2
3
4
5
6
```

Sample Output 0

```
2D distance = 5
3D distance = 6
```

Explanation

For the first point: x=1, y=2, z=3.
For second point: x=3, y=5, z=6.

2D distance using the formula = 4.242640687119285 = 5 (rounding up the result).
3D distance using the formula = 5.196152422706632 = 6 (rounding up the result).

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour.

Start tour

Original code | Java 7

```
1 import ↔;
4
5
6
```

```
7  //Enter your code here
8  class Point2D {
9      public int x;
10     public int y;
```

VMWare Challenge UR Propel

28m : 31s
to test end

1

2

3

4

```
15     }
16
17     public double dist2D(Point2D p) {
18         double distance = Math.sqrt((p.x - x) * (p.x - x) +
   (p.y - y) * (p.y - y));
19         return distance;
20     }
21
22     public void printDistance(double d) {
23         System.out.print("2D distance = ");
24         if (d > (int) d) {
25             System.out.println((int) d + 1);
26         } else {
27             System.out.println((int) d);
28         }
29     }
30 }
31
32
33 class Point3D extends Point2D {
34     public int z;
35
36     Point3D(int x, int y, int z) {
37         super(x, y);
38         this.z = z;
39     }
40
41     public double dist3D(Point3D p) {
42         double dist2d = dist2D(p);
43         double dist3d = Math.sqrt(dist2d * dist2d + (z - p.z)
   * (z - p.z));
44
45         return dist3d;
46     }
47
48     public void printDistance(double d) {
49         System.out.print("3D distance = ");
50         if (d > (int) d) {
51             System.out.println((int) d + 1);
52         } else {
53             System.out.println((int) d);
```

```
54          }
55      }
56  }
```

```
57 public class Solution {↔}
```

VMWare Challenge UR Propel

28m : 31s
to test end

Test against custom input

Run Code | Submit code & Continue

(You can submit any number of times)

1

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

2

3

4

Compiled successfully. All available test cases passed!

Test Case #1:	✔	Test Case #6:	✔
Test Case #2:	✔	Test Case #7:	✔
Test Case #3:	✔	Test Case #8:	✔
Test Case #4:	✔	Test Case #9:	✔
Test Case #5:	✔		

Testcase 1: *Success*

Your Output

```
2D distance = 5
3D distance = 6
```

Expected Output

```
2D distance = 5
3D distance = 6
```

Testcase 2: *Success*

Your Output

```
Output hidden
```

Testcase 3: *Success*

Your Output

```
Output hidden
```

Testcase 4: *Success*

Your Output

```
Output hidden
```

Testcase 5: *Success*

Your Output

```
Output hidden
```

Testcase 6: *Success*

Your Output

```
Output hidden
```

Testcase 7: *Success*

Your Output

```
Output hidden
```

Testcase 8: *Success*

Your Output

```
Output hidden
```

Testcase 9: *Success*

Your Output

```
Output hidden
```